Computação I - Python
Aula 3 - Teórica: Tipos de dados, Strings, Estrutura Condicional

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ

Tipos de Dados - Dados Numéricos

Em computação, um tipo de dado é uma classificação dos dados. Essa classificação determina como os dados serão armazenados e quais operações já definidas na linguagem de programação serão aplicáveis.

- **Números Inteiros**: int
- **Ponto Flutuante**: float
- **Números Complexos**: complex

- Operações com dados de um mesmo tipo tendem a gerar resultados do mesmo tipo dos operandos.
- Operações com dados de diferentes tipos geram resultados do tipo mais complexo.

Tipos de Dados - Sequência de caracteres: str

- Constantes string são escritas usando aspas simples ou duplas

  "a" ou 'a'

- O operador + pode ser usado para concatenar strings

  "a"+"b" é o mesmo que "ab"

- O operador * pode ser usado para repetir strings

  "a"*10 é o mesmo que "aaaaaaaaaa"

Tipos de Dados - Sequência de caracteres: str

Podemos usar a função **soma(x,y)** para concatenar strings ?

```
1 def soma(x,y):
2     """Esta e a funcao soma que dados os valores de x e y
      retorna o valor de x + y"""
3     return x+y
```

Tipos de Dados - Sequência de caracteres: str

Podemos usar a função **soma(x,y)** para concatenar strings ?

```
1 def soma(x,y):
2     """Esta e a funcao soma que dados os valores de x e y
      retorna o valor de x + y"""
3     return x+y
```

```
1 >>> soma("lady","bug")
2     'ladybug'
```

Tipos de Dados - Sequência de caracteres: str

Podemos usar a função **soma(x,y)** para concatenar strings ?

```
1 def soma(x,y):
2     """Esta e a funcao soma que dados os valores de x e y
      retorna o valor de x + y"""
3     return x+y
```

```
1 >>> soma(15,' anos')
2 Traceback (most recent call last):
3   File "<pyshell#1>", line 1, in <module>
4     soma(15,'anos')
5   File "/home/joao/Desktop/soma.py", line 6, in soma
6     return x+y
7 TypeError: unsupported operand type(s) for +: 'int' and 'str'
8
9 >>> soma('15',' anos')
10    '15 anos'
```

Conversão entre tipos de dados

- Dados numéricos *não* são convertidos automaticamente para o tipo string

```
>>> soma('15',' anos')
    '15 anos'

>>> "Minha idade e "+ 15 +" anos"
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    "Minha idade e "+ 15 + " anos."
TypeError: cannot concatenate 'str' and 'int' objects

>>> "Minha idade e "+ str(15) +" anos"
    "Minha idade e 15 anos"
```

- Para converter uma string em inteiro ou float podemos usar:

```
>>> int("15")
    15

>>> float("3.14")
    3.14
```

String

Escreva uma função que receba como parâmetro o nome e a idade de uma pessoa, e que retorne a frase:

"Olá fulano, meu nome é Python e eu tenho x anos. "

onde *fulano* e *x* são, respectivamente, o nome e o dobro da idade do usuário.

String

Escreva uma função que receba como parâmetro o nome e a idade de uma pessoa, e que retorne a frase:

"Olá fulano, meu nome é Python e eu tenho x anos. "

onde *fulano* e *x* são, respectivamente, o nome e o dobro da idade do usuário.

```
1 def olafulano(nome,idade):
2
3   """Funcao que recebe nome e idade e escreve uma frase.
4   Os parametros de entrada sao do tipo (str,int).
5   O valor de retorno e do tipo (str)"""
6
7   return "Ola "+ nome +", meu nome e Python, e tenho " + str
      (2*idade) + " anos."
```

Tipos de Dados - Booleano (bool)

- Assume apenas dois valores: verdadeiro (True) ou falso (False)
- É o tipo de dado resultante das operações de comparação.

```
>>> 3>2
    True

>>> 10 <= 5
    False
```

Relações e Expressões Booleanas

- Operadores: $>$, $<$, $==$ (igual), $!=$ (diferente), $>=$, $<=$

ATENÇÃO

- $X == Y$: operador relacional $\Rightarrow$ X É IGUAL A Y
- $X = Y$: operador de atribuição $\Rightarrow$ ATRIBUIR A X O VALOR DE Y

Relações e Expressões Booleanas

- **Relações**: $>$, $<$, $==$ (igual), $!=$ (diferente), $>=$, $<=$
- **Operadores Lógicos**: not (negação), and (e), or (ou)
- **Expressões Booleanas**: Retornam como resultado de sua avaliação os valores verdadeiro (True) ou falso (False)

Exp 1	Exp 2	Exp 1 and Exp 2	Exp 1 or Exp 2	not Exp 1
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Exp1 e **Exp2** podem ser dados booleanas ou podem ser expressões booleanas compostas de operadores relacionais e operadores lógicos.

Relações e Expressões Booleanas

Ordem de Precedência: do maior para o de menor precedência

1. $**$
2. $*, /, //, \%$
3. $+, -$
4. $<, <=, >, >=, !=, ==$
5. **not**
6. **and**
7. **or**

Expressões Booleanas

Faça uma função booleana que retorne **True** caso o número passado como parâmetro seja par, e **False** caso contrário.

Expressões Booleanas

Faça uma função booleana que retorne **True** caso o número passado como parâmetro seja par, e **False** caso contrário.

```
1 def par(numero):
2
3   """Funcao Booleana que retorna True quando passamos um
      numero par.
4   Parametro de Entrada: int
5   Valor de Retorno : bool"""
6
7   return numero%2 == 0
```

Expressões Booleanas

Faça uma função booleana que retorne **True** caso o número passado como parâmetro seja par, e **False** caso contrário.

```
1 def par(numero):
2
3   """Funcao Booleana que retorna True quando passamos um
      numero par.
4   Parametro de Entrada: int
5   Valor de Retorno : bool"""
6
7   return numero%2 == 0
```

```
1 >>> par(6)
2     True
3
4 >>> par(1)
5     False
```

Expressões Booleanas

Faça uma função booleana que retorne **True** caso o número passado como parâmetro seja ímpar, e **False** caso contrário. Use a função *par(numero)* definida antes.

```
1 def par(numero):
2
3   """Funcao Booleana que retorna True quando passamos um numero par.
4   Parametro de Entrada: int
5   Valor de Retorno : bool"""
6
7   return numero%2 == 0
```

Expressões Booleanas

Faça uma função booleana que retorne **True** caso o número passado como parâmetro seja ímpar, e **False** caso contrário. Use a função *par(numero)* definida antes.

```
1 def par(numero):
2
3   """Funcao Booleana que retorna True quando passamos um numero par.
4   Parametro de Entrada: int
5   Valor de Retorno : bool"""
6
7   return numero%2 == 0
```

```
1 def impar(numero):
2
3   """Funcao Booleana que retorna True quando passamos um numero impar.
4   Parametro de Entrada: int
5   Valor de Retorno : bool"""
6
7   return not par(numero)
```

Expressões Booleanas

Calcule o resultado das seguintes expressões, sabendo que:

- os parâmetros A e B são inteiros e valem 2 e 7
- o parâmetro C é do tipo float e vale 3.5
- o parâmetro L é booleano e vale *False*.

1. $(2 < 5)$ and $((15/3) = 5)$
2. $(2 < 5)$ and $((15/3) == 5)$
3. $B = A * C$ and $(L$ or $True)$
4. $B == A * C$ and $(L$ or $True)$
5. not L or $True$ and $(A + B >= C)$
6. $((B/A) == C)$ or $((B/A)! = C)$
7. $((B/A) == C)$ or $((B/A)! = C)$ and L

Expressões Booleanas

Calcule o resultado das seguintes expressões, sabendo que:

- os parâmetros A e B são inteiros e valem 2 e 7
- o parâmetro C é do tipo float e vale 3.5
- o parâmetro L é booleano e vale *False*.

1. $(2 < 5)$ and $((15/3) = 5)$: SyntaxError: invalid syntax
2. $(2 < 5)$ and $((15/3) == 5)$: True
3. $B = A * C$ and $(L$ or $True)$: B passa a ser True
4. $B == A * C$ and $(L$ or $True)$: True
5. not L or $True$ and $(A + B >= C)$: True
6. $((B/A) == C)$ or $((B/A)! = C)$: True
7. $((B/A) == C)$ or $((B/A)! = C)$ and L : False

Expressões Booleanas

Defina uma função booleana que dada uma idade retorna *True* se a idade for maior ou igual a 18 e *False* caso contrário.

Expressões Booleanas

Defina uma função booleana que dada uma idade retorna *True* se a idade for maior ou igual a 18 e *False* caso contrário.

```
1 def maiorDeIdade(idade):
2   """Funcao booleana que dada uma idade retorna True se a idade for maior ou igual a 18 e
        False caso contrario.
3   Parametro de Entrada: int
4   Valor de Retorno : bool"""
5
6   return idade >=18
```

Defina uma função booleana que dados 3 números retorna *True* se eles formam um triângulo equilatero e *False* caso contrário.

Expressões Booleanas

Defina uma função booleana que dada uma idade retorna *True* se a idade for maior ou igual a 18 e *False* caso contrário.

```
1 def maiorDeIdade(idade):
2   """Funcao booleana que dada uma idade retorna True se a idade for maior ou igual a 18 e
        False caso contrario.
3   Parametro de Entrada: int
4   Valor de Retorno : bool"""
5
6   return idade >=18
```

Defina uma função booleana que dados 3 números retorna *True* se eles formam um triângulo equilatero e *False* caso contrário.

```
1 def triEquilatero(A,B,C):
2
3   """Funcao booleana que dados 3 numeros retorna True se eles formam um triangulo
        equilatero e False caso contrario.
4   Parametros de Entrada: float,float,float
5   Valor de Retorno : bool"""
6
7   return A == B == C
```

Expressões Booleanas

Defina uma função booleana que dados 3 números retorna *True* se eles formam um triângulo isoseles e *False* caso contrário.

Expressões Booleanas

Defina uma função booleana que dados 3 números retorna *True* se eles formam um triângulo isoseles e *False* caso contrário.

```
1 def trIsosceles(A,B,C):
2
3   """Funcao booleana que dados 3 numeros retorna True se eles formam um triangulo
        isoseles e False caso contrario.
4   Parametros de Entrada: float,float,float
5   Valor de Retorno : bool"""
6
7   return (A == B) or (A == C) or (C == B)
```

Expressões Booleanas

Defina uma função booleana que dados 3 números retorna *True* se eles formam um triângulo isoseles e *False* caso contrário.

```
1 def trilsosceles(A,B,C):
2
3   """Funcao booleana que dados 3 numeros retorna True se eles formam um triangulo
        isoseles e False caso contrario.
4   Parametros de Entrada: float,float,float
5   Valor de Retorno : bool"""
6
7   return (A==B) or (A==C) or (C==B)
```

```
1 >>> trilsosceles(3.0,3.0,2.0)
2     True
3 >>> trilsosceles(3.0,3.0,3.0)
4     True
```

Expressões Booleanas

Defina uma função booleana que dados 3 números retorna *True* se eles formam um triângulo isoseles e *False* caso contrário.

```
1 def trilsosceles(A,B,C):
2
3   """Funcao booleana que dados 3 numeros retorna True se eles formam um triangulo
        isoseles e False caso contrario.
4   Parametros de Entrada: float,float,float
5   Valor de Retorno : bool"""
6
7   return (A==B) or (A==C) or (C==B)
```

```
1 >>> trilsosceles(3.0,3.0,2.0)
2     True
3 >>> trilsosceles(3.0,3.0,3.0)
4     True
```

O que fizemos errado?

Expressões Booleanas

Defina uma função booleana que dados 3 números retorna *True* se eles formam um triângulo isoseles e *False* caso contrário.

```
1 def trilsosceles(A,B,C):
2
3   """Funcao booleana que dados 3 numeros retorna True se eles formam um triangulo
        isoseles e False caso contrario.
4   Parametros de Entrada: float,float,float
5   Valor de Retorno : bool"""
6
7   return (A==B) or (A==C) or (C==B)
```

```
1 >>> trilsosceles(3.0,3.0,2.0)
2     True
3 >>> trilsosceles(3.0,3.0,3.0)
4     True
```

O que fizemos errado?

```
1 def trilsosceles(A,B,C)
2   """Funcao booleana ..."""
3   return ((A==B) and (A!=C)) or ((A==C) and (A!=B)) or ((C==B) and (A!=C))
```

```
1 def trilsosceles(A,B,C)
2   """Funcao booleana ..."""
3   return ((A==B) or (A==C) or (C==B)) and (not (A==B==C))
```

Expressões Booleanas

```
1 def trilsosceles(A,B,C)
2   """Funcao booleana ..."""
3   return ((A == B) and (A != C)) or ((A == C) and (A != B)) or ((C == B) and (A != C))
```

```
1 def trilsosceles(A,B,C)
2   """Funcao booleana ..."""
3   return ((A == B) or (A == C) or (C == B)) and (not (A == B == C))
```

```
1 def trilsosceles(A,B,C)
2
3   """Funcao booleana ..."""
4
5   return ((A == B) or (A == C) or (C == B)) and (not triEquilatero(A,B,C))
```

Expressões Booleanas

Defina uma função booleana que dado um número (inteiro ou float) retorna *True* se ele for maior ou igual a zero e *False* caso contrário.

Expressões Booleanas

Defina uma função booleana que dado um número (inteiro ou float) retorna *True* se ele for maior ou igual a zero e *False* caso contrário.

```
1 def PositivoBool(n)
2
3   """Funcao booleana que dado um numero retorna True se ele for
      maior ou igual a zero e False caso contrario.
4   Parametro de Entrada: int ou float
5   Valor de Retorno : bool"""
6
7   return n >= 0
```

Expressões Booleanas

Defina uma função booleana que dado um número (inteiro ou float) retorna *True* se ele for maior ou igual a zero e *False* caso contrário.

```
1 def PositivoBool(n)
2
3   """Funcao booleana que dado um numero retorna True se ele for
      maior ou igual a zero e False caso contrario.
4   Parametro de Entrada: int ou float
5   Valor de Retorno : bool"""
6
7   return n >= 0
```

```
1 >>> PositivoBool(5)
2     True
3 >>> PositivoBool(0)
4     True
5 >>> PositivoBool(-7.0)
6     False
```

Estrutura Condicional

Faça uma função que, dado um número inteiro X passado como parâmetro, retorna a string *"X é positivo"* caso X seja um número positivo, e *"X não é positivo"* caso contrário.

Estrutura Condicional

Faça uma função que, dado um número inteiro X passado como parâmetro, retorna a string *"X é positivo"* caso X seja um número positivo, e *"X não é positivo"* caso contrário.

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

Estrutura Condicional

Faça uma função que, dado um número inteiro X passado como parâmetro, retorna a string *"X é positivo"* caso X seja um número positivo, e *"X não é positivo"* caso contrário.

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

Estrutura Condicional

```
if < expressão >:
    < comandos 1 >
else:
    < comandos 2 >
```

Estrutura Condicional

Faça uma função que, dado um número inteiro X passado como parâmetro, retorna a string *"X é positivo"* caso X seja um número positivo, e *"X não é positivo"* caso contrário.

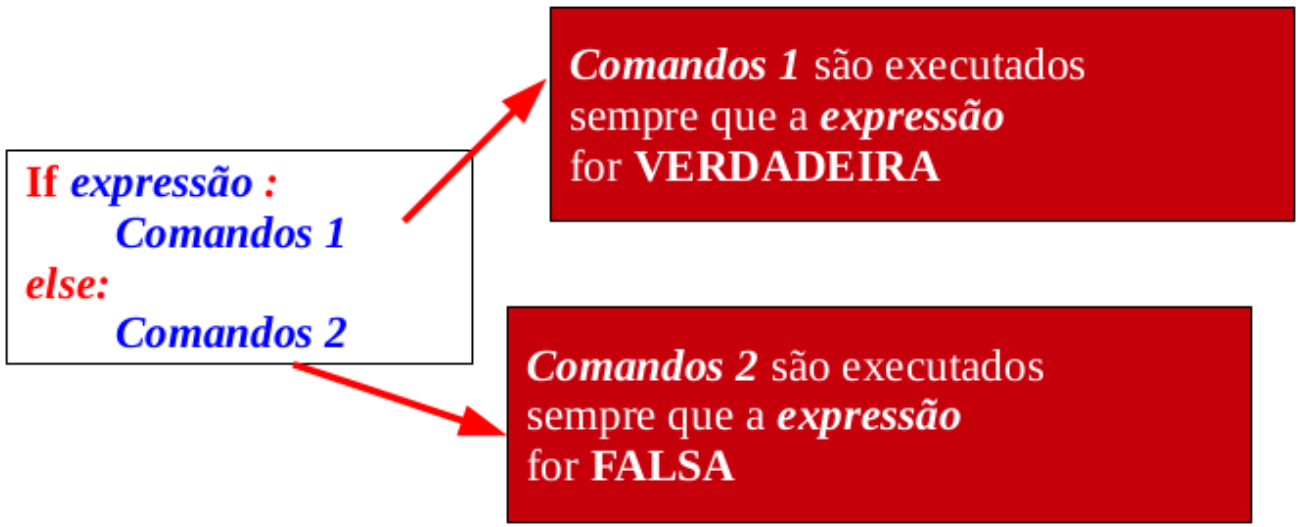
```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

Estrutura Condicional

```
if < expressão >:
    < comandos 1 >
else:
    < comandos 2 >
```

A **expressão** na estrutura condicional é
do tipo booleano - verdadeira (True) ou falsa (False).

Estrutura Condicional Composta



Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(3)
```

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(3)
```

positivo(3):

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(3)
```

positivo(3):
    if 3 > 0 :

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(3)
```

positivo(3):

```
if 3 > 0 :
    return str(3) + ' e positivo'
else:
    return str(X) + ' nao e positivo'
```

```
'3 e positivo'
```

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(-5)
```

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(-5)
```

positivo(-5):

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(-5)
```

positivo(-5):
    if -5 > 0 :

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(-5)
```

positivo(-5):

```
    if -5 > 0 :
        return str(X) + 'e positivo'
    else:
        return str(X) + 'nao e positivo'
```

```
'-5 nao e positivo'
```

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(0)
```

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(0)
```

positivo(0):

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(0)
```

positivo(0):
    if 0 > 0 :

Estrutura Condicional

```
1 def positivo(X):
2   """Funcao que recebe um numero inteiro e determina se ele e positivo.
3   Parametro de Entrada: int
4   Valor de Retorno : str"""
5
6   if X > 0 :
7     return str(X) + " e positivo"
8   else:
9     return str(X) + " nao e positivo"
```

```
>>> positivo(0)
```

positivo(0):

```
if 0 > 0 :
    return str(X) + 'e positivo'
else:
    return str(X) + 'nao e positivo'
```

```
'0 nao e positivo'
```

Estrutura Condicional

Faça uma função que determina se um número inteiro X passado como parâmetro é positivo, negativo ou zero. O valor de retorno da função deve ser uma dentre as strings *"X é positivo"*, *"X é negativo"* ou *"X é zero"*.

Estrutura Condicional

Faça uma função que determina se um número inteiro X passado como parâmetro é positivo, negativo ou zero. O valor de retorno da função deve ser uma dentre as strings *"X é positivo"*, *"X é negativo"* ou *"X é zero"*.

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(0)
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(0)
```

def *PosNegZero(0):*

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(0)
```

def *PosNegZero(0):*
    if 0 > 0 :

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(0)
```

```
def PosNegZero(0):
    if 0 > 0 :
        return str(X) + ' e positivo'
    else:
        if 0 < 0 :
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(0)
```

```
def PosNegZero(0):
    if 0 > 0 :
        return str(X) + ' e positivo'
    else:
        if 0 < 0 :
            return str(X) + ' e negativo'
        else:
            return str(0) + 'e zero'
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(0)
```

```
def PosNegZero(0):
    if 0 > 0 :
        return str(X) + ' e positivo'
    else:
        if 0 < 0 :
            return str(X) + ' e negativo'
        else:
            return str(0) + 'e zero'
```

```
'0 e zero'
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(2)
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(2)
```

def *PosNegZero(2):*

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(2)
```

def *PosNegZero(2):*
    if 2 > 0 :

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(2)
```

```
def PosNegZero(2):
    if 2 > 0 :
        return str(2) + ' e positivo'
    else:
        if X < 0 :
            return str(X) + ' e negativo'
        else:
            return str(X) + 'e zero'
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(2)
```

```
def PosNegZero(2):
    if 2 > 0 :
        return str(2) + ' e positivo'
    else:
        if X < 0 :
            return str(X) + ' e negativo'
        else:
            return str(X) + 'e zero'
```

```
'2 e positivo'
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(-5)
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(-5)
```

def *PosNegZero(-5):*

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(-5)
```

def *PosNegZero(-5):*
    if -5 > 0 :

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(-5)
```

```
def PosNegZero(-5):
    if -5 > 0 :
        return str(X) + ' e positivo'
    else:
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(-5)
```

```
def PosNegZero(-5):
    if -5 > 0 :
        return str(X) + ' e positivo'
    else:
        if -5 < 0 :
```

Estrutura Condicional

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
>>> PosNegZero(-5)
```

```
def PosNegZero(-5):
    if -5 > 0 :
        return str(X) + ' e positivo'
    else:
        if -5 < 0 :
            return str(-5) + ' e negativo'
        else:
            return str(X) + 'e zero'
```

```
'-5 e negativo'
```

Estrutura Condicional

else: ... if ⇔ elif ...:

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    else:
7      if X < 0 :
8        return str(X) + " e negativo"
9      else:
10       return str(X) + " e zero"
```

```
1  def PosNegZero(X):
2    """Funcao ..."""
3
4    if X > 0 :
5      return str(X) + " e positivo"
6    elif X < 0 :  # ESTA LINHA MUDOU !
7      return str(X) + " e negativo"
8    else:
9      return str(X) + " e zero"
```

Estrutura Condicional

Faça uma função que dada a data de nascimento de uma pessoa, retorna sua idade. Caso a pessoa esteja fazendo aniversário, além da idade, deve ser retornado uma mensagem de parabéns.

Dica: Para saber a data atual, use o módulo *datetime* e a função *datetime.datetime.now()*.

```
>>> import datetime
>>> datetime.datetime.now()
datetime.datetime(2015, 6, 10, 18, 52, 58, 305960)
>>> datetime.datetime.now().year
    2015
>>> datetime.datetime.now().month
    6
>>> datetime.datetime.now().day
    10
>>> datetime.datetime.now().hour
    18
>>> datetime.datetime.now().minute
    53
>>> datetime.datetime.now().second
    21
>>> datetime.datetime.now().microsecond
    151031
```

Estrutura Condicional

Faça uma função que dada a data de nascimento de uma pessoa, retorna uma string que informe a idade do usuário e uma mensagem de parabéns caso ele esteja fazendo aniversário.

Estrutura Condicional

Faça uma função que dada a data de nascimento de uma pessoa, retorna uma string que informe a idade do usuário e uma mensagem de parabéns caso ele esteja fazendo aniversário.

```
1  import datetime
2  def idade(dia,mes,ano):
3    """Funcao que calcula a idade de uma pessoa.
4    Parametros de Entrada: int, int, int
5    Valor de Retorno : str"""
6
7    if dia == datetime.datetime.now().day and mes == datetime.datetime.now().month:
8
9      return str(datetime.datetime.now().year - ano) + """ anos. Parabens pelo aniversario
         """
10
11   elif ((mes < datetime.datetime.now().month) or
12 (mes == datetime.datetime.now().month and dia < datetime.datetime.now().day)):
13
14     return str(datetime.datetime.now().year - ano) + """ anos."""
15
16   else:
17
18     return str(datetime.datetime.now().year - ano - 1) + """ anos."""
```

Estrutura Condicional

1. Faça uma função que receba como entrada o código de uma mercadoria e o preço e retorne como saída o preço da mercadoria, sendo que se o código for '00' um desconto de 10% no preço deve ser aplicado.

2. Faça uma função que receba como entrada dois números e retorne o maior deles. Os valores são, por definição, diferentes entre si.

3. Faça uma função que receba como entrada dois números e retorne o maior deles. Caso os números sejam iguais, retorne *"Os números são iguais"*.

Estrutura Condicional

4. Faça uma função que receba como entrada **três** números e retorne o maior deles. Caso os três números sejam iguais, retorne *"Os números são iguais"*.Faça o chinês da sua função para as seguintes entradas:

Entrada	Valor de Retorno
(92541 , 7.432 , -1)	?
(9.1 , 9.1 , 5L)	?
(-5.2 , 0 , 10)	?

5. Uma faculdade atribui menções aos alunos conforme a faixa de notas que tenha atingido:
   - 9,0 a 10: S (superior)
   - 7,0 a 8,9: MS (médio superior)
   - 5,0 a 6,9: M (médio)
   - 0,0 a 4,9: MI (médio inferior)

   Faça uma função que dada a nota retorna a menção. Caso a nota não esteja entre 0 e 10, retornar mensagem de *"nota inválida"*.

Estrutura Condicional

6. Faça uma função em Python que receba como entrada:

   - as notas P1, P2 e P3 das provas, e
   - a nota da avaliação prática AP de um aluno de computação 1.

   Sua função deve então retornar a média deste aluno e uma string dizendo *"aprovado"* caso o aluno tenha sido aprovado, ou *"reprovado"* caso contrário.

Autores

- João C. P. da Silva ▸ Lattes
- Carla Delgado ▸ Lattes
- Ana Luisa Duboc ▸ Lattes

Colaboradores

- Anamaria Martins Moreira ▸ Lattes
- Fabio Mascarenhas ▸ Lattes
- Leonardo de Oliveira Carvalho ▸ Lattes
- Charles Figueiredo de Barros ▸ Lattes
- Fabrício Firmino de Faria ▸ Lattes

Computação I - Python
Aula 3 - Teórica: Tipos de dados, Strings, Estrutura Condicional

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ